QT: multi-platform library for API and GUI *(a brief introduction)*

VGG Seminar, 1.3.2006

Matúš Straka

AUSTRIAN ACADEMY OF SCIENCES
ÖSTERREICHISCHE AKADEMIE DER WISSENSCHAFTEN

Part I – QT Introduction

Part II – How things in QT work

Part III – Useful QT classes

Applications & Operating Systems

- OS is a basic equipment of a computer - applications are built within given OS
- Currently used OSes:
  - MS Windows (XP, 2000, 9x, ME, NT)
  - Linux/Unix with X11 (HP-UX, Solaris, …)
  - Apple Mac OS X
  - Embedded: Embedded Linux, MS Windows Mobile, Symbian, …
- OS differ in:
  - Architecture (API, GUI) + hardware
  - Licensing + costs

What is API and GUI?

- API – Application Program Interface:
  - Set of routines, protocols and tools for building an application (e.g. provided by OS)
  - Serves for HW abstraction, memory/ disk/ video/ network routines, clip-boards, printing, …
- GUI – Graphical User Interface
  - Set of ‘widgets’ available for programmer to build interface between user and applications
  - windows, buttons, lists, check-boxes, …

Multi-platform Development

- It is preferable:
  - Applications not bound to given HW or OS
  - Applications can be used even if particular OS disappears
  - Potentially larger group of users
- It is NOT easy:
  - Memory architecture is given by HW and OS
  - OS API and GUI elements differ
    - Different names, parameters, approaches
    - Some might be missing (e.g. STL, …)
  - Differences acceptable for non-GUI applications
  - #ifdef statements not feasible
- Available solutions:
  - Applications has to be written in unified GUI and API

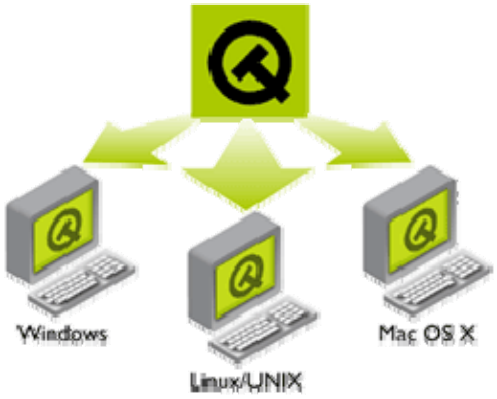
  - Trolltech QT, wxWindows, GTK+, …

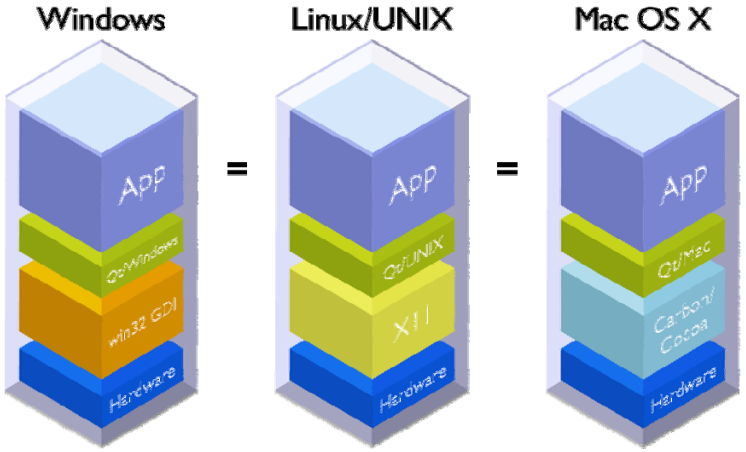
Trolltech's QT

- Creator: Trolltech, Norway, since 1994
- Cross-platform library for Win, Unix/Linux, Mac
- Offers replacement for:
  - **GUI:** MFC, Aqua-feel in OSX, Motif, is a base of KDE, …
  - **API:** STL (string, vector, list, map, …)
- Simplifies:
  - Disk/file operations, multi-threading
  - Date/time, registers/ini/.rc settings saving
  - GDI graphics, OpenGL usage

How Does QT Work?

- QT is native on Win/Linux/Max
  - Written down to the lowest level
  - No emulation
  - Fast and robust
- QT applications:
  - Native, compiled executables
  - No interpretation





Part I – QT Introduction

Part II – How things in QT work

Part III – Useful QT classes

QT Basics – Event Based UIs

- Modern UIs are usually event-based
- Event types:
  - User actions (key press, mouse click/move, …)
  - Timers, system events (“new mail arrived”), …
- One event can trigger other events
- Application might trigger own events
- Application has to respond to events!
- How to connect event “senders” and “recipients”?

QT Basics – SIGNALs and SLOTs

- QT terminology:
  - SIGNAL = event, emit SIGNAL = create event
  - SLOT = event processor
- SIGNAL/SLOT is a mechanism to bind event “senders” and “recipients”
- SIGNALs/SLOTs are macros internally recompiled to event-loop structures (*moc* files)
- Syntax:
  - *CONNECT(sender, signal, receiver, slot)*

QT Basics – SIGNALs/SLOTs

```
class Army{
   Q_OBJECT
public:
   Army(…);
   ~Army();
   void battle()
   {
         emit command(true)
   }
signals:
   void command(bool now);
}

class Soldier
{
public:
   Soldier(…);
   ~Soldier(…);
public slots:
   void getUp(bool now)
         { openEyes(); }
}
```

```
void main()
{
   Army smallArmy;
   Soldier youngSoldier;
   float budget = 1e10;

   connect(smallArmy,
          SIGNAL(command(bool)),
          youngSoldier,
          SLOT(getUp(bool)));

   while(budget-- > 0)
   {
        army->battle();
                // soldiers will get up
   }
}
```

Part I – QT Introduction

Part II – How things in QT work

Part III – Useful QT classes

Useful QT classes

- Also in computer graphics, effective applications with good UI are needed
- Typical tasks:
  - general: strings
  - filename, file/dir, disk operations (selection)
  - data-processing: vectors, lists, maps
  - images (JPEG, PNG, …)
  - graphics (GDI graphics, OpenGL, …)
  - Simple HTML browser
  - XML
  - Support for OS independent multi-threading

QString

- A class for handling character strings
- QString str:
  - str.contains()
  - str.lower(), str.upper()
  - str = number(n), n = str.toInt()
  - char* s = str.latin1()

QVector, QValueList, QMap classes

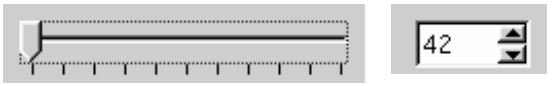
- Template classes (e.g. QMap<int, QString>)
- Iterators and [] operators available
- In QT3.x – STL like iterators
- In QT4 – *foreach(item, container)* approach
- Easier and more friendly than STL counterparts

QImage

- Bitmap images processing
- 2-8-16-32 bit images
- Up to 4000x4000
- QImage img:
  - img(fileName)
  - img.save(fileName, “format”)
  - v = img.pixel(x, y), img.setPixel(x, y, v)
  - img.scale(newW, newH)

QWidgets

- User interface widgets:
  - QLabel
  - QPushButton, QRadioButton
  - QCheckBox
  - QLineEdit
  - QTextEdit (formatting)
  - QSlider, QSpinBox
  - QListBox, QListView, QComboBox, …
  - QLayout – organizing widgets within window
    - Used with QDesigner tool



QPainter, QGLWidget

- QPainter – GDI graphics
  - vector graphics for screen and printer
  - drawLine(), drawRectangle(), drawPolygon(), …
- QGLWidget
  - OpenGL canvas
  - After makeCurrent() call standard OpenGL commands can be used for given window

QThread

- Simple platform independent multi-threading

```
class MyThread : public QThread {
public:
      virtual void run();
    };

void MyThread::run()
{
     for( int count = 0; count < 20; count++ ) {
            sleep( 1 );
            qDebug( "Ping!" );
     }
}
int main()
{
   MyThread a;
   MyThread b;
   a.start();
   b.start();
   a.wait();
   b.wait();
}
```

QT Support Tools

- In Windows, QT is integrated with MS Visual C++ (6.0, .net)
- **QDesigner**: design&layout of windows and dialogs
- **QLinguist**: translation of application to other languages
- **QAssistant**: hypertext documentation and help
- 20+ **tutorials/demos** (with growing complexity, very instructive)

QT Licence Policy

- Dual Licence Policy:
- **Open Source**
  - QT4 fully open source (even for Windows), but bound to MinGW compiler (MinGW is also open source)
  - You **have to** make you application Open Source also!
- **Closed Source** – academic/educational/commercial
  - Integration with Visual Studio, …
  - License is '*per-developer*' (not per computer)
  - Educational – only schools and their HW
  - Academic – also non-profit research organizations
  - Academic and Educational Licences cannot be used for commercially sold/leased/rented products

QT Qualities

- Very good user-interface, good help
- Fast and stable

- Adobe Photoshop Elements
- Skype, Google Earth,
- IBM, KDE

- *QT* and *f3d* shall fullfill **ALL** volume graphics programmer needs (maybe *xisl* and some math library (e.g. *NumRecipes*) needed)

TROLLTECH®

http://www.trolltech.com

That’s all, folks …

matus.straka@oeaw.ac.at